EFFICIENT MESSAGE-PASSING FOR DISTRIBUTED QUADRATIC OPTIMIZATION

Guoqiang Zhang and Richard Heusdens

Department of Intelligent Systems
Delft University of Technology
Delft, the Netherlands
{g.zhang-1,r.heusdens}@tudelft.nl

ABSTRACT

Distributed quadratic optimization (DQO) has found many applications in computer science and engineering. In designing a message-passing algorithm for DQO, the basic idea is to decompose the quadratic function into a set of local functions with respect to a graphic model. The nodes in the graph send local information of the quadratic function in message-form to their neighbors iteratively until reaching the global optimal solution. The efficiency of a message-passing algorithm depends on its computational complexity, the number of parameters to be transmitted, and its convergence speed. In this work, we study several message-passing algorithms for comparison. In particular, we consider the Jacobi-relaxation algorithm, the generalized linear coordinate descent (GLiCD) algorithm and the min-sum-min algorithm.

1. INTRODUCTION

In this work, we consider solving the quadratic optimization problem in a distributed fashion, namely

$$\min_{x\in\mathbb{R}^n} f(x) = \min_{x\in\mathbb{R}^n} \left(\frac{1}{2}x^\top Jx - h^\top x\right), \qquad (1)$$

where the quadratic matrix J is real symmetric positive definite and x is a real vector in n-dimensional space. It is known that the optimal solution is given by $x^* = J^{-1}h$. We suppose that the quadratic matrix J is sparse and the dimensionality n is large. In this situation, the direct computation (without using the sparse structure of J) of the optimal solution may be expensive and unscalable. The research challenge is how to exploit the sparse geometry of J to efficiently obtain the optimal solution.

A common approach that exploits the sparsity of J is to associate the function $f(x)$ with an undirected graph $G = (V, E)$. That is, the graph has a node for each variable x_i and an edge between node i and j only if the element J_{ij} is nonzero. By doing so, the sparsity of J is fully captured by the graph. As a consequence, the function can be decomposed with respect to $G = (V, E)$ as

$$f(x) = \sum_{i\in V} f_i(x_i) + \sum_{(i,j)\in E} f_{ij}(x_i, x_j), \qquad (2)$$

where each edge-function $f_{ij}(x_i, x_j)$ characterizes the interaction of x_i and x_j as specified by J_{ij}. With the graphic model (2), distributed quadratic optimization (DQO) boils down to how to spread the global information of (J, h) in (1) over the graph efficiently by exchanging local information between neighboring nodes.

DQO over graphic models has found many applications in computer science and engineering in the past. Some applications are motivated by emerging parallel computational architectures (e.g., multicore CPUs and GPUs [1]), such as support vector machine [2] and channel coding [3, 4]. Other applications are motivated from the distributed nature carried by the problem, such as distributed speech enhancement in wireless microphone networks [5], distributed Kalman filter [6] and multiuser detection[7].

2. ALGORITHM COMPARISON

In the literature, the Jacobi algorithm is a classic method for solving the quadratic problem over the associated graph [8]. At each iteration, the algorithm performs node-oriented minimizations over all the nodes in the graph, of which the messages are in a form of linear functions (see Table 1). It is known that when the matrix J is walk-summable[1], the Jacobi algorithm converges to the optimal solution [9, 10]. To fix the convergence for a general matrix J, the Jacobi algorithm was under-relaxed by incorporating an estimate of x^* from last iteration in computing a new estimate (see Table 1). It is well known that the Jacobi-relaxation algorithm possesses a guaranteed convergence if the relaxation parameter is properly chosen [8]. For the above two algorithms, once a node-estimate is updated, this estimate is broadcast to all its neighbors. Because the information transmitted is general, and not edge-specific, the two algorithms are known to converge slowly [8].

To accelerate the convergence of the Jacobi algorithm, we proposed the linear coordinate descent (LiCD) algorithm [11]. At each iteration, the LiCD algorithm performs pairwise minimizations over all the edges in the graph, of which the messages are in a form of linear functions (see Table 1). As shown in [11], if the quadratic matrix J is walk-summable, the LiCD algorithm converges to the optimal solution. Inspired by the Jacobi-relaxation

[1] A positive definite matrix $J \in \mathbb{R}^{n\times n}$, with all ones on its diagonal, is walk-summable if the spectral radius of the matrix $\bar{R}$, where $R = I - J$ and $\bar{R} = [|R_{ij}|]_{i,j=1}^n$, is less than one (i.e., $\rho(\bar{R}) < 1$). We note that if the matrix J is diagonally dominant, it is also walk-summable.

J is walk-summable	J is general
Jacobi Alg.: * node-oriented minimization * linear message	Jacobi-relaxation Alg.: * introduce feedback in Jacobi Alg.
LiCD Alg.: * pairwise minimization * linear message	GLiCD Alg.: * introduce feedback in LiCD Alg.
min-sum Alg.: * pairwise minimization * quadratic message	min-sum-min Alg.: * introduce feedback in min-sum Alg.

Table 1. Algorithm comparison.

algorithm, we also extended the LiCD algorithm by incorporating feedback from last iteration in computing new messages in [12]. We name the new algorithm as the *generalized LiCD* (GLiCD) algorithm. The GLiCD algorithm was shown in [12] to converge to the optimal solution for a general matrix J when the amount of feedback signal is set to be large enough. For both the LiCD and the GLiCD algorithms, each node computes and transmits edge-specific information instead of broadcasting some common parameters to all its neighbors. Such edge-specific operation helps to spread the global information of (J, h) over the graph more effectively.

An alternative scheme for solving the quadratic problem is by using the framework of probability theory [13]. The optimal solution x^* is viewed as the mean value of a random vector $x \in \mathbb{R}^n$ with Gaussian distribution

$$p(x) \propto \exp\left(-\frac{1}{2}x^\top Jx + h^\top x\right). \tag{3}$$

The min-sum algorithm is one popular approach to estimate both the mean value $x^* = J^{-1}h$ and individual variances [14]. At each iteration, the algorithm essentially performs pairwise minimizations over all the edges in the graph, of which the messages are in a form of quadratic functions (see Table 1). For a graph with a tree-structure, the min-sum algorithm converges to the optimal solution in finite steps [14]. The question of convergence for loopy graphic models has been proven difficult. In [9, 10], it was shown when the matrix J is walk-summable, the min-sum algorithm converges to the optimal solution.

Due to the fact that the min-sum algorithm may fail a general matrix J, we proposed the min-sum-min algorithm [15] recently. The derivation of the min-sum-min algorithm follows the line of work in [12] for the GLiCD algorithm. Similarly to the GLiCD algorithm, the basic idea of the min-sum-min algorithm is to incorporate feedback from last iteration in computing new messages. We have shown in [15] that if the amount of the feedback is large enough, the min-sum-min algorithms converges to the optimal solution. We note that for the min-sum and the min-sum-min algorithms, each node computes and transmits edge-specific information to its neighbors, which is similar to that of the LiCD and the GLiCD algorithms.

The main properties of the above algorithms are summarized in Table 1. One observes that the Jacobi and the LiCD algorithms share the property that their messages are in the form of linear functions. On the other hand, the LiCD and the min-sum algorithms share the property that both algorithms perform pairwise minimization at each iteration. From the viewpoint of minimization strategies and message-forms, the LiCD algorithm acts as an intermediate method between the Jacobi and the min-sum algorithms. As is analyzed in [11], the computational complexities of the three algorithms at each iteration are in the order of

$$\text{Jacobi Alg.} \rightarrow \text{LiCD Alg.} \rightarrow \text{min-sum Alg.}$$

where the min-sum algorithm is most expensive for implementation.

3. UNIFIED MESSAGE-PASSING FRAMEWORK

We note that all the algorithms listed in Table 1 share a unified message-passing framework despite the fact that different minimization strategies and message-forms are applied in the algorithms. We present the unified message-passing framework in the following.

Consider the quadratic optimization problem (1). We may assume, without loss of generality, that J is of unit-diagonal. The local node and edge functions for the graph $G = (V, E)$ can be constructed as

$$f_i(x_i) = \frac{1}{2}x_i^2 - h_i x_i \qquad i \in V \tag{4}$$

$$f_{ij}(x_i, x_j) = J_{ij}x_i x_j \qquad (i,j) \in E. \tag{5}$$

An edge exists between node i and j in the graph only if $J_{ij} \neq 0$. As a consequence, a sparse matrix J leads to a sparse graph $G = (V, E)$. We use $N(i)$ to denote the set of all neighbors of node $i \in V$. The set $N(i)\backslash j$ excludes the node j from $N(i)$. For each edge $(i, j) \in E$, we use $[j, i]$ and $[i, j]$ to denote its two directed edges. Correspondingly, we denote the set of all directed edges of the graph as $\vec{E}$.

A message-passing algorithm exchanges information between neighboring nodes iteratively until reaching consensus. In particular, at time t, each node j collects a set of messages $\left\{m_{v\to j}^{(t)}(x_j)|v \in N(j)\right\}$ and a set of estimates $\left\{\hat{x}_{j|v}^{(t)}, v \in N(j)\right\}$ of x_j^* by cooperating with its neighbors. We note that for a directed edge $[v, j] \in \vec{E}$, the

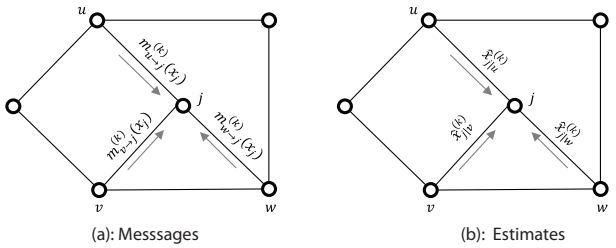

Figure 1. An example of the information-flow for node j at time step k.

associated message $m_{v\to j}^{(t)}(x_i)$ and estimate $\hat{x}_{j|v}$ are obtained by combining the local information of node v and j at time $t-1$ (see Fig. 1). For the Jacobi and the Jacobi-relaxation algorithms, the elements in $\{\hat{x}_{j|v}^{(t)}, v \in N(j)\}$ for each node j are identical, since both algorithms perform node-oriented minimizations.

Given the messages at time t, one can define new local functions as

$$f_i^{(t)}(x_i) = f_i(x_i) + \sum_{u\in N(i)} m_{u\to i}^{(t)}(x_i) \quad i \in V$$

$$f_{ij}^{(t)}(x_i, x_j) = \Big[f_{ij}(x_i, x_j) - m_{j\to i}^{(t)}(x_i) - m_{i\to j}^{(t)}(x_j)\Big] \quad (i,j) \in E$$

By summing up all the new local functions, it is straightforward that

$$f(x) = \sum_{i\in V} f_i^{(t)}(x_i) + \sum_{(i,j)\in E} f_{ij}^{(t)}(x_i, x_j). \qquad (6)$$

Thus, the overall objective function remains the same. The new local functions can be viewed as a reformulation of the objective function.

The key part of a message-passing algorithm is the derivation of the updating expressions for $\big\{(m_{j\to i}^{(t+1)}(x_i), \hat{x}_{i|j}^{(t+1)}), [j,i] \in \vec{E}\big\}$ given the information at time t. Note that for each node i, the estimates $\{\hat{x}_{i|u}^{(t)}, u \in N(i)\}$ provide information about the optimal solution x_i^*. Thus, the estimates can be used as feedback in computing new messages and estimates in next iteration if necessary. An iterative algorithm converges to the optimal solution x^* if

$$\lim_{t\to\infty} \hat{x}_{i|j}^{(t)} = x_i^*, [j,i] \in \vec{E}. \qquad (7)$$

Different iterative algorithms can be derived by choosing different minimization strategies and message-forms (see Table 1). As an example, we briefly present the Jacobi-relaxation algorithm in the following for demonstration. At time t, each node i keeps track of an estimate $\hat{x}_i^{(t)}$ of x_i^* and a set of linear messages $\{m_{u\to i}^{(t)}(x_i) = J_{iu}\hat{x}_u^{(t)}\}$. The estimate $\hat{x}_i^{(t+1)}$ at time step $t+1$ is computed as [8]

$$\hat{x}_i^{(t+1)} = \min_{x_i} \Big[f_i^{(t)}(x_i) + \frac{\alpha}{2}(x_i - \hat{x}_i^{(t)})^2\Big] \quad i \in V, (8)$$

where the parameter $\alpha \in \mathbb{R}$ controls the amount of feedback in computing $\hat{x}_i^{(t+1)}$. Note that the feedback in (8) is represented by a quadratic penalty function in terms of $\hat{x}_i^{(t)}$, which can be easily merged into the local function $f_i^{(t)}(x_i)$. By letting $\alpha = 1 - \frac{1}{s}$, the above expression can be reformulated as

$$\hat{x}_i^{(t+1)} = \min_{x_i} \Big[sf_i(x_i) + \sum_{u\in N(i)} sJ_{ui}\hat{x}_u^{(t)} + \frac{1-s}{2}(x_i - \hat{x}_i^{(t)})^2\Big] \quad i \in V.$$

In the literature, s is named as the relaxation parameter. When $s = 1$ (or equivalently, $\alpha = 0$), the Jacobi-relaxation algorithm reduces to the Jacobi algorithm. For a general matrix J in (1), the Jacobi-relaxation algorithm converges to the optimal solution x^* if the relaxation parameter s is sufficiently close to zero from above.

For those who are interested in the GLiCD and the min-sum-min algorithms, we refer the readers to [12] and [15]. Similarly to that of the Jacobi-relaxation, the feedbacks in the GLiCD and the min-sum-min algorithms are also represented by some quadratic penalty functions. The amount of feedback signal in the GLiCD algorithm or the min-sum-min algorithm is again controlled by a relaxation parameter.

4. FUTURE WORK

We note that the Jacobi and the Jacobi-relaxation algorithms have a wide range of applications in practice. Naturally, it is worth trying other algorithms as listed in Table 1 for solving the same kind of problems. In future work, we will consider applying the GLiCD algorithm the min-sum-min algorithms for some practical problems.

5. REFERENCES

[1] Y. El-Kurdi, W. J. Gross, and D. Giannacopoulos, "Efficient implementation of gaussin belief propagation solver for large sparse diagonally domiant linear systems," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 471–474, 2012.

[2] D. Bickson, D. Dolev, and E. Yom-Tov, "A Gaussian belief propagation solver for large scale Support Vector Machines," in *5th European Conference on Complex Systems*, Sept. 2008.

[3] H. Uchikawa, B. M. Kurkoski, K. Kasai, and K. Sakaniwa, "Iterative Encoding with Gauss-Seidel Method for Spatially-Coupled Low-Density Lattice Codes," in *Proc. IEEE Int. Symp. Information Thoery*, MIT Campus, USA, 2012.

[4] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Trans. Information Theory*, vol. 54, pp. 15611585, Apr. 2008.

[5] R. Heusdens, G. Zhang, R. C. Hendriks, Y. Zeng, and W. B. Kleijn, "Distributed MVDR Beamforming for (Wireless) Microphone Networks Using

Message Passing," accepted by *International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2012.

[6] D. Bickson, O. Shental, and D. Dolev, "Distributed Kalman Filter via Gaussian Belief Propagation," in *the 46th Allerton Conf. on Communications, Control and Computing*, 2008.

[7] D. Bickson, O. Shental, P. H. Siegel, J. K. Wolf, and D. Dolev, "DGaussian belief propagation based multiuser detection," in *In IEEE Int. Symp. on Inform. Theory (ISIT)*, July 2008, pp. 1878–1882.

[8] D. P. Bertsekas and J. N. Tsitsikis, *Parallel and distributed Computation: Numerical Methods*, Belmont, MA: Athena Scientific, 1997.

[9] J. K. Johnson, D. M. Malioutov, and A. S. Willsky, "Walk-sum Interpretation and Analysis of Gaussian Belief Propagation," in *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, 2006, vol. 18.

[10] D. M. Malioutov, J. K. Johnson, and A. S. Willsky, "Walk-Sums and Belief Propagation in Gaussian Graphical Models," *J. Mach. Learn. Res.*, vol. 7, pp. 2031–2064, 2006.

[11] G. Zhang and R. Heusdens, "Linear Coordinate-Descent Message-Passing for Quadratic Optimization," appering in Neural Computation.

[12] G. Zhang and R. Heusdens, "Convergence of Generalized Linear Coordinate-Descent Message-Passing for Quadratic Optimization," in *Proc. IEEE International Symposium on Information Theory*, June 2012.

[13] S.L. Lauritzen, *Graphical Models*, Oxford University Press, 1996.

[14] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," *Morgan Kaufman Publishers*, 1988.

[15] G. Zhang and R. Heusdens, "Convergence of Min-Sum-Min Message-Passing for Quadratic Optimization," in preparation for submission.